Name : Shahriar Ahmed
ID: 20101588
Section : 11
Faculty Initial : SEJ
Exam : Semester Final.

Question-1:

Create a dictionary that stores Fibonacci values up to m where the key will be n(n>=1) and the value will be the nth Fibonacci. Fibonacci series will start with 0,1. At first, the user will input the value of m, followed by 3 queries where the user will input n and you have to print the nth Fibonacci number. If the n given in the query is not present in the dictionary, print 'No fibo'.

Sample Input:
20
3
5
11
Sample Output:
1
3
No fibo

Answer:

```
nterms = int(input())
a=int(input())
b=int(input())
c=int(input())
dict={}
n1=0
n2=1
count = 0
i=1
while n1 < nterms:
   dict[i] = n1
   nth = n1 + n2
   n1 = n2
   n2 = nth
   count += 1
   i += 1
if dict.get(a)!=None:
```

```
    print(dict.get(a))
else:
    print("No fibo")
if dict.get(b)!=None:
    print(dict.get(b))
else:
    print("No fibo")
if dict.get(c)!=None:
    print(dict.get(c))
else:
    print("No fibo")
```

Question-2:

Implement the design of the **Avengers** class so that the following output is produced:

Driver Code:

```
# Write your code here

print('Total Avengers:', Avengers.count)
a1 = Avengers('Captain America', 'Bucky Barnes')
a1.super_powers('Stamina', 'Slowed ageing')
a2 = Avengers('Doctor Strange', 'Ancient One')
a2.super_powers('Mastery of magic', 'Gifted surgeon')
a3 = Avengers('Iron Man', 'War Machine')
a3.super_powers('Genius level intellect', 'Scientist ')
print("=========================")
a1.printAvengersDetail()
print("=========================")
a2.printAvengersDetail()
print("=========================")
a3.printAvengersDetail()
print("=========================")
print('Total Avengers:', Avengers.count)
```

Output:

```
Total Avengers: 0
=========================
Name: Captain America
Partner: Bucky Barnes
Super powers: Stamina , Slowed ageing
=========================
Name: Doctor Strange
Partner: Ancient One
Super powers: Mastery of magic , Gifted surgeon
=========================
Name: Iron Man
Partner: War Machine
Super powers: Genius level intellect , Scientist
=========================
Total Avengers: 3
```

Answer:

```
class Avengers:
    count = globals()
    count = 0
    def __init__(self,name,partner):
        self.name = name
        self.partner = partner

    def super_powers(self, a, b):
        self.super_power1 = a
        self.super_power2 = b

    def printAvengersDetail(self):
        print ("Name: ", self.name,
               "\nPartner: ", self.partner,
               "\nSuper powers: ",self.super_power1,self.super_power2)
        Avengers.count += 1

print('Total Avengers:', Avengers.count)
a1 = Avengers('Captain America', 'Bucky Barnes')
a1.super_powers('Stamina', 'Slowed ageing')
a2 = Avengers('Doctor Strange', 'Ancient One')
a2.super_powers('Mastery of magic', 'Gifted surgeon')
a3 = Avengers('Iron Man', 'War Machine')
a3.super_powers('Genius level intellect', 'Scientist ')
```

```
print("========================")
a1.printAvengersDetail()
print("========================")
a2.printAvengersDetail()
print("========================")
a3.printAvengersDetail()
print("========================")
print('Total Avengers:', Avengers.count)
```

Question-3:

Implement the design of the **Asus** and the **Dell** classes that are derived from the **Laptop** class so that the following code generates the output below:

Driver Code:

```
class Laptop:
    def __init__(self, name):
        self.name = name
    def check():
        print("The Laptop is working properly")

# Write your code here

a = Asus("Zenbook 14")
d = Dell("Inspiron")
a.check()
print("========================")
d.check()
print("========================")
```

Output:

```
This is Asus Zenbook 14
The Laptop is working properly
========================
This is Dell Inspiron
The Laptop is working properly
========================
```

Answer:

```
class Laptop:

    def __init__(self, name):
        self.name = name
    def check(self):
        print(f"This is {self.__class__.__name__} {self.name}")
        print("The Laptop is working properly")

class Asus(Laptop):

    def __init__(self,name):
        super().__init__(name)

class Dell(Laptop):

    def __init__(self,name):
        super().__init__(name)

a = Asus("Zenbook 14")
d = Dell("Inspiron")
a.check()
print("=========================")
d.check()
print("=========================")
```

Question-4:

Create a dictionary that stores the following series values up until K defined by the user input. Key is the position 👇, which will start from 1 and increase by 1 every time. The value for each key will be its corresponding series value of nth position. Print the dictionary.

Finally, you need to take another input based on which you will print whether that number is part of the following series or not. If it is part of the series, print the key and value pair. Otherwise, print ‘no such value exists’
To solve this task, you need to take 2 inputs.
The first input, K which defines when to end calculating the series.
The second input, Val which will be used to check whether the input is part of the series or not.
Series: 1, 5, 12, 22, 35, 51, ….., K (K must be less than or equal to the

user input.)

The equation to calculate the series: ((3 * n^2)- n)/2, n starts from 1 and increases by 1

Sample Input 1:
37
12
Sample Output 1:
{1:1, 2:5, 3:12, 4:22, 5:35}
Key: 3, Value: 12

Sample Input 2:
51
25
Sample Output 2:
{1:1, 2:5, 3:12, 4:22, 5:35, 6:51}
no such value exists

Answer:

```
k=int(input())
lol=int(input())
n=1
i=1
dict={}
c=0
while True:
  c=((3 * i^2 )- i)/2
  dict[i]=int(((3 * i*i )- i)/2)
  if dict[i] >k:
     dict.pop(i)
     break
  else:
     i+=1
print(dict)
for name in dict:
  if dict[name]==lol:
     print("Key:",name)
     print("Value",lol)
     exit()
print("No such value exist")
```

Question-5:

Create a dictionary that stores the following series values up until the K defined by the user input. Key is the position 👎, which will start from 1 and increase by 1 every time. The value for each key will be its corresponding series value of nth position. Print the dictionary.
Finally, you need to take another input based on which you will print whether that number is part of the following series or not. If it is part of the series, print the key and value pair. Otherwise, print 'no such value exists'

To solve this task, you need to take 2 inputs.
The first input, K which defines when to end calculating the series.
The second input, Val which will be used to check whether the input is part of the series or not.

Series: 1, 6, 15, 28, 45, 66, ….., K (K must be less than or equal to the user input.)

Equation to calculate the series: (2n * (2n- 1))/2, n starts from 1 and increases by 1 every time.

Sample Input 1:
65
28
Sample Output 1:
{1:1, 2:6, 3:15, 4:28, 5:45}
Key: 4, Value: 28

Sample Input 2:
28
11
Sample Output 2:
{1:1, 2:6, 3:15, 4:28}
no such value exists

Answer:

```
k=int(input())
lol=int(input())
n=1
i=1
dict={}
c=0
```

```
while True:
  c=((3 * i^2 )- i)/2
  dict[i]=int((2*i * (2*i- 1))/2)
  if dict[i] >k:
     dict.pop(i)
     break
  else:
     i+=1
print(dict)
for name in dict:
  if dict[name]==lol:
     print("Key:",name)
     print("Value",lol)
     exit()
print("No such value exist")
```

Question-6:

Implement the design of the **Account** class so that the following output is produced:

Driver Code:

```
# Write your code here

print('No of account holders:', Account.count)
print("=========================")
p1 = Account("Abdul", 45, "Service Holder", 500000)
p1.addMoney(300000)
p1.printDetails()
print("=========================")
p2 = Account("Rahim", 55, "Businessman", 700000)
p2.withdrawMoney(700000)
p2.printDetails()
print("=========================")
p3 = Account("Ashraf", 62, "Govt. Officer", 200000)
p3.withdrawMoney(250000)
p3.printDetails()
print("=========================")
print('No of account holders:', Account.count)
```

Output:

```
No of account holders: 0
=========================
Add Money successfully !!
```

```
Name: Abdul
Age:  45
Occupation:  Service Holder
Total Amount:  800000
==========================
Withdraw Successful !!
Name: Rahim
Age:  55
Occupation:  Businessman
Total Amount:  0
==========================
Not sufficient balance.
Name: Ashraf
Age:  62
Occupation:  Govt. Officer
Total Amount:  200000
==========================
No of account holders: 3
```

Answer:

```
class Account:
  count=0
  def __init__(self,name,age,prof,bal):
      self.name=name
      self.age=age
      self.profession=prof
      self.balance=bal
      Account.count+=1
  def addMoney(self,n):
      self.balance+=n
      print("Add Money successfully!!")
  def printDetails(self):
      print(f"Name: {self.name}")
      print(f"Age: {self.age}")
      print(f"Occupation: {self.profession}")
      print(f"Total Amount:{self.balance}")
  def withdrawMoney(self,n):
      if n>self.balance:
          print("Not sufficient balance")
      else:
          self.balance-=n
          print("Withdraw Successful!!")
print('No of account holders:', Account.count)
```

```
print("=========================")
p1 = Account("Abdul", 45, "Service Holder", 500000)
p1.addMoney(300000)
p1.printDetails()
print("=========================")
p2 = Account("Rahim", 55, "Businessman", 700000)
p2.withdrawMoney(700000)
p2.printDetails()
print("=========================")
p3 = Account("Ashraf", 62, "Govt. Officer", 200000)
p3.withdrawMoney(250000)
p3.printDetails()
print("=========================")
print('No of account holders:', Account.count)
```

Question-7:

Implement the **Attacker** and the **Defender** class that inherit from **Player** class so that the following code generates the output below:
[N.B: Point is increased by 2 times the player rating]

Driver Code:

```
class Player:
    def __init__(self,name,goalsScored,tacklesWon):
        self.name = name
        self.goalsScored = goalsScored
        self.tacklesWon = tacklesWon
        self.point=0
    def calculatePoint(self):
        self.point+=(self.goalsScored*4)+(self.tacklesWon*3)


# Write your code here

print('=========================')
p1 = Defender("Thiago Silva",5,12,8.5)
print('=========================')
p2 = Attacker("Cristiano Ronaldo",14,5,9.0)
print('=========================')
p3 = Attacker("Lionel Messi",12,9,9.5)
print('=========================')
p1.calculatePoint()
print('=========================')
p2.calculatePoint()
print('=========================')
```

p3.calculatePoint()

Output:

```
========================
Name: Thiago Silva ,Rating: 8.5
========================
Name: Cristiano Ronaldo ,Rating: 9.0
========================
Name: Lionel Messi ,Rating: 9.5
========================
Point of  Thiago Silva : 73.0
========================
Point of  Cristiano Ronaldo : 89.0
========================
Point of  Lionel Messi : 94.0
```

Answer:

```
class Player:
  def __init__(self,name,goalsScored,tacklesWon):
      self.name = name
      self.goalsScored = goalsScored
      self.tacklesWon = tacklesWon
      self.point=0
  def calculatePoint(self):
      self.point+=(self.goalsScored*4)+(self.tacklesWon*3)
      if self.point<=60:
          self.point+=17
      elif self.point<=74:
          self.point+=18
      else:
          self.point+=19
      print("point of ",self.name,":",self.point)

class Attacker(Player):
  def __init__(self,name,goalsScored,tacklesWon,rat):
      super().__init__(name,goalsScored,tacklesWon)
      self.rat=rat
      print(self.name,",",self.rat)
class Defender(Player):
  def __init__(self,name,goalsScored,tacklesWon,rat):
      super().__init__(name,goalsScored,tacklesWon)
      self.rat=rat
      print(self.name,",",self.rat)
print('========================')
p1 = Defender("Thiago Silva",5,12,8.5)
```

```
print('=========================')
p2 = Attacker("Cristiano Ronaldo",14,5,9.0)
print('=========================')
p3 = Attacker("Lionel Messi",12,9,9.5)
print('=========================')
p1.calculatePoint()
print('=========================')
p2.calculatePoint()
print('=========================')
p3.calculatePoint()
```

Question-8:

Implement the design of the **FootBallTeam** and the **CricketTeam** classes that inherit from **Team** class so that the following code generates the output below:

Driver Code:

```
class Team:
    def __init__(self, name):
        self.name = name
        self.total_player = 5
    def info(self):
        print("We love sports")
# Write your code here.

class Team_test:
    def check(self, tm):
        print("=========================")
        print("Total Player:", tm.total_player)
        tm.info()

f = FootBallTeam("Brazil")
c = CricketTeam("Bangladesh")
test = Team_test()
test.check(f)
test.check(c)
```

Output:

```
=========================
Total Player: 11
Our name is Brazil
```

We play Football
We love sports
==========
Total Player: 11
Our name is Bangladesh
We play Cricket
We love sports

Answer:

```
class Team:
  def __init__(self, name, sports):
      self.name = name
      self.total_player = 5
      self.sports = sports
  def info(self):
      print('Our name is {} '.format(self.name))
      print("We play {} ".format(self.sports))
      print('We love sports')


class FootBallTeam(Team):
  def __init__(self,name):
      self.name = name
      self.total_player = 11
      self.sports = "Football"

class CricketTeam(Team):
  def __init__(self,name):
      self.name = name
      self.total_player = 11
      self.sports = "Cricket"


class Team_test:
  def check(self, tm):
      print("==========")
      print('Total Player:', tm.total_player)
      tm.info()

f = FootBallTeam('Brazil')
c = CricketTeam('Bangladesh')
test = Team_test()
test.check(f)
```

test.check(c)

Question-9:

Suppose a dictionary contains the marks of each student where key is the student name and value is the mark. Your task is to print the name of the pair of students whose average mark is equal or greater than 90. For example, a dict contains the following elements

My_dict = {‘A’:87,’B’:93,’C’:’90,’D’:84,’E’:89}
Pair of students having 90 or more marks as average are - AB, BC, BE.
BA, CB and EB should not be printed as the pairs are printed once.

Sample Input:
{‘A’:87,’B’:93,’C’:’90,’D’:84,’E’:89}
Sample Output:
AB
BC
BE

Answer:

```
a=input()
a=a.replace("{","")
a=a.replace("}","")
a=a.replace("'","",a.count("'"))
a=a.replace(":",",",a.count(":"))
a=a.split(",")
b={}
for i in range(0,len(a),2):
    b[a[i]]=int(a[i+1])
for i in range(0,len(a)-3,2):
    for j in range(i+2,len(a),2):
        if (b[a[i]]+b[a[j]])/2 >=90:
            print(a[i]+a[j])
```

Question-10:

Implement the design of the **Patient** class so that the following output is produced:

Driver Code:

 # Write your code here

```
print(“Total Patient:”, Patient.count)
p1 = Patient(“Thomas”, 23)
p1.add_Symptom(“Fever”, “Headache”)
p2 = Patient(“Carol”, 20)
p2.add_Symptom(“Vomiting”, “Coughing”)
p3 = Patient(“Mike”, 25)
p3.add_Symptom(“Fever”, “Coughing”)
print("==========================")
p1.printPatientDetail()
print("==========================")
p2.printPatientDetail()
print("==========================")
p3.printPatientDetail()
print("==========================")
print(“Total Patient:”, Patient.count)
```

Output:

```
Total Patient: 0
==========================
Name: Thomas
Age: 23
Symptoms: Fever, Headache
==========================
Name: Carol
Age: 20
Symptoms: Vomiting, Coughing
==========================
Name: Mike
Age: 25
Symptoms: Fever, Coughing
==========================
Total Patient: 3
```

Answer:

```
class Patient: count = 0
    def __init__(self, name, age):
        self.name = name
        self.age = age
        Patient.count += 1
    def add_Symptom(self, *info):
        self.lt = [val for val in info]
```

```
    def printPatientDetail(self):
        print(f'Name: {self.name}')
        print(f'Age: {self.age}')
        print(f'Sysptoms: {self.lt[0]},{self.lt[1]}')

print('Total Patient:', Patient.count)
p1 = Patient('Thomas', 23)
p1.add_Symptom('Fever', 'Headache')
p2 = Patient('Carol', 20)
p2.add_Symptom('Vomiting', 'Coughing')
p3 = Patient('Mike', 25)
p3.add_Symptom('Fever', 'Coughing')
print("=========================")
p1.printPatientDetail()
print("=========================")
p2.printPatientDetail()
print("=========================")
p3.printPatientDetail()
print("=========================")
print('Total Patient:', Patient.count)
```

Question-11:

Implement the design of the **FootBallTeam** and the **CricketTeam** classes that inherit from **Team** class so that the following code generates the output below:

Driver Code:

```
class Team:
    def __init__(self, name):
        self.name = name
        self.total_player = 5
    def info(self):
        print("We love sports")
# Write your code here.

class Team_test:
    def check(self, tm):
        print("=========================")
        print("Total Player:", tm.total_player)
        tm.info()

f = FootBallTeam("Brazil")
```

```
c = CricketTeam(“Bangladesh”)
test = Team_test()
test.check(f)
test.check(c)
```

Output:

```
========================
Total Player: 11
Our name is Brazil
We play Football
We love sports
========================
Total Player: 11
Our name is Bangladesh
We play Cricket
We love sports
```

Answer:

```
class Team:
    def __init__(self, name):
        self.name = name
        self.total_player = 5
    def info(self):
        print("We love sports")

class FootBallTeam(Team):
    def __init__(self, name):
        self.name = name
        self.total_player = 11
    def info(self):
        print("Our name is",self.name)
        print("We play Football")
        print("We love sports")

class CricketTeam(Team):
    def __init__(self, name):
        self.name = name
        self.total_player = 11
    def info(self):
        print("Our name is",self.name)
        print("We play Cricket")
        print("We love sports")
```

```
class Team_test:
    def check(self, tm):
        print("=========================")
        print("Total Player:", tm.total_player)
        tm.info()

f = FootBallTeam("Brazil")
c = CricketTeam("Bangladesh")
test = Team_test()
test.check(f)
test.check(c)
```